Module 42
A315b, Part 2 of 2: Specifying Requirements for Actuated Traffic Signal Controllers (ASC) Based on NTCIP 1202 v03 Standard

Ken Leonard: ITS Standards can make your life easier. Your procurements will go more smoothly and you'll encourage competition, but only if you know how to write them into your specifications and test them. This module is one in a series that covers practical applications for acquiring and testing standards-based ITS systems.

I am Ken Leonard, director of the ITS Joint Program Office for USDOT and I want to welcome you to our newly redesigned ITS standards training program of which this module is a part. We are pleased to be working with our partner, the Institute of Transportation Engineers, to deliver this new approach to training that combines web-based modules with instructor interaction to bring the latest in ITS learning to busy professionals like yourself.

This combined approach allows interested professionals to schedule training at your convenience without the need to travel. After you complete this training, we hope that you will tell colleagues and customers about the latest ITS standards and encourage them to take advantage of the archived version of the webinars.

ITS Standards training is one of the first offerings of our updated Professional Capacity Training Program. Through the PCB program, we prepare professionals to adopt proven and emerging ITS technologies that will make surface transportation safer, smarter, and greener which improves livability for us all. You can find information on additional modules and training programs on our website www.pcb.its.dot.gov.

Please help us make even more improvements to our training modules through the evaluation process. We look forward to hearing your comments. Thank you again for participating and we hope you find this module helpful.

Kenneth Vaughn: Hi. Welcome to Module A315b Part 2. This is for Specifying Requirements for Actuated Traffic Signal Controllers based on the NTCIP 1202 v03 Standard. This is Module Part 2 of 2. The previous module went through the various segments of how you look at requirements and implement those. This course will focus more on some of the details more intimate to that specification. And this module was updated in July 2020.

I'll be your presenter today, Kenneth Vaughn. I'm the President of Trevilon LLC and I've been working with NTCIP standards since the 1990s, and have been testing and developing systems engineering process, also working closely with the architecture and connected vehicles here in the U.S. and internationally. So some learning objectives for today's course includes managing special considerations for this particular standard NTCIP 1202, both related to infrastructure and functionality. Once we go through all of those, we'll look at how we might incorporate requirements that are not currently supported by the standard and give some examples of those. And then finally, we'll close out by just doing a brief introduction to testing. There will be a future testing module.

So with the learning objectives, Managing Special Considerations related to infrastructure. We will go over some key items here today. So we're going to talk a little bit about, kind of, the history of NTCIP and how that has led to a particular set of design characteristics that we developed with the NTCIP design and how that might impact your specification today. Specifically, to do that we need to consider the simple network management protocol, SNMP,

which is kind of the driving engine behind NTCIP. And why we developed the STMP or the simple transportation management protocol. And then also consider exception based reporting and block objects, two other ways to reduce the overhead consumption of traditional SNMP. And then finally, we'll talk some about infrastructure limitations in communications loadings that you should consider as you write your specification.

So with that, let's consider the origins of NTCIP. Back in 1993 when we first started developing NTCIP standards, it was a very different world than what we see in the field today. The goal was still the same, was to support remote communications, although at the time it was really just for traffic signal control. NTCIP started primarily for traffic signal controllers, other devices were added a couple years later. It was intended to be the communications protocol for NEMA TS 2, which was traffic controller assemblies. So originally intended for signal control but then it quickly expanded to support a range of other devices within the ITS realm.

It was also designed with that traffic signal control of the 1990s; there was recognition that there was a variety of different architectures for controlling traffic signals. That included central control on a second by second basis. It included lots of field masters controlling their signals as well as truly distributed control. And whatever protocol we developed at the time needed to support all of those different environments. In addition, even though we are having central control on a second by second basis, a lot of those systems were based on 1200 BPS or bits per second environment. Yes, merely 1K bit per second environment.

There's recognition that higher speeds would emerge. Clearly, at the time, the 1990s, we did have much higher speeds for our computers, but out in the field we had multi-drop copper cables that really would only support 1200 bps. Greater innovations got it up to maybe 9600 in cases, but that was still a real limitation of these copper wires that were installed in the field and that existing infrastructure and the cost replacement infrastructure stayed with us a long time. And that it may have in many cases started out with central control or field masters but it stayed with us throughout even when we moved to distributed control, that same infrastructure was still in place.

But all these different characteristics led to a very varied environment. We had a frequency of command in monitoring messages that were very different. Central control often had second by second commands, whereas closed loop systems had perhaps less frequent commands whereas distributed controls had even less frequent commands. The content of the command also varied across these different platforms that all related to the operational structure what data needed to be exchanged. Then finally, the number of devices sharing a particular communications medium also varied and once you start putting on multiple signals on a 1200 bps circuit, that really starts limiting your ability to communicate very much information.

Well, the good news is when we started development of NTCIP, we looked at existing standards of how professionals develop communications protocols. And dating back to the seventies was this OSI stack, open systems interconnect model, and it was a 7 layer defined stack by International Standard Organization, ISO, that identified the key functions that had to be performed when you define communications. So at the physical layer, you have to define what a logical bit true and false is and how that's exchanged across a wire or over the air. The data link decided how I package those bits together to form data from point A to point B, you know, from one device to another device.

The network layer talked about how I transfer that message not only from point A to point B but from point A to point Z on the network. And then the transport layers said, “Well, I'm not only interested in transmitting individual messages or individual packets of information.” Those packets might need to be grouped together to form a larger message and that's what the transport layer does. The session layer then allows me to create multiple messages back and forth as a session. The presentation talks about how logical data from the application gets encoded. And then the application layer talks about what the actual message and content is.

The NTCIP, when we started out, we developed our model based on that but we simplified things a little bit because we realized that existing standards grouped things together logically. And for example, Data Link standards where coupled, tended to be good for certain types of physical networks. They're still independent standards but they tend logical pairings would occur together. So that's what we did is we created a Subnet that combined data link and physical. We created a transport that combined the transport layer and the network layer. We created an application that combined application presentation and session.

In particular at that level, things get very muddled. They keep going, kind of, back and forth on what services they do, where and how those stacks—how those layers arrange with each other. And then finally, we also got into, kind of, the end application or what we call the information layer. And at this layer, we talk about just defining the data independent of how it's exchanged. So we actually added a layer on top to cover that aspect as well and that's actually a lot of where the NTCIP standards are, particularly the 1200 series documents including 1202, the Standard we'll focus on today.

Now since that model was developed in the nineties, TC 204 of ISO is developed. That's the Intelligent Transport Systems Committee within the International Standards Organization. They develop what's called the ITS station architecture and that architecture actually parallels—if you look in the middle there— parallels almost exactly what the NTCIP did. Slightly different names but they have an access layers, which is the exact parallel to the NTCIP subnet layer. There's a Transnet layers, which is an exact parallel of the transport layer of NTCIP. Then we have a facility layer, which is the same as the application layer of NTCIP.

And then finally, we have this thing called an application entity on top of that that roughly parallels the information layer of NTCIP but it goes a bit beyond that and it allows you to specify a lot of other details and performance characteristics and other things about the application, the thing you actually want to do with the communications environment. Then you'll see two other entities that are very important. One is the management entity. That management entity allows me to control characteristics of each one of those layers. So if you think at the subnet layer or access layer in the case of ITS station, I might need to control my data rate over my COM link. That's what the management layer does.

The other key aspect is the security entity. The security entity is distinct and separate because security has to be considered all the way throughout and that entity is providing services to everything else. So traditionally, people kind of thought of security as just kind of fitting in to the transport layer or somewhere else; we now look at that as an independent entity that manages everything. This is key, because this allows a device to be authenticated at the Transnet layer and then that authentication information is stored in the security entity and it can be referenced by the application entity to dictate access control, a very key security concept there. That's

where the industry is headed. Most of NTCIP right now is focused on this more middle layer and that's what we'll focus on in this course.

So with that, let's look at the first protocol developed by the NTCIP, which was not developed by but referenced by, adopted by the NTCIP, which was the internet standard Simple Network Management Protocol, SNMP. It is, as I said, a major internet standard. It provides a very flexible message structure, which is what allowed us to adopt it and then customize our own data. And the benefit of this approach is that the manager decides when to send each message and what data to contain within that message.

So let's take a look at this a little bit closer. How does this actually work? The NTCIP 1202, for example, defines its standardized objects. All of the 1200 series NTCIP documents, that's what they focus on. One standard object as defined is phase Status Group Green. It reports what phases are currently green, one bit per phase. So the value 34 indicates its 2 and 6 are on. Actually, it's bit 1 and 5, but they mapped it to phase 2 and 6. And then that gets encoded to the 34. There's another for alarm status that just reports bits for different types of alarms. We'll assume there's no alarm right now.

So the NTCIP standard defines both of those objects and defines what their values mean. And SMNP then couples these into the way I share my information is by identifying the name of the object and I follow that by its value and then I can have multiple objects paired together so then I can have another name followed by its value, another name followed by its value, et cetera.

So that's the information layer and that gets embedded into this SNMP packet. And that packet then also includes other content. It includes a version number. It includes a community name, giving me a certain level of access rights. It indicates the type of message it is. In this case, we're going to assume it's a response message coming from the device. And the request message obviously wouldn't be giving values to the device, it would just be saying please give me this information. It includes a message ID. It includes error codes. So in this case there's no errors in my response. I'm getting the value I want back. And that's what the SNMP does.

And then that SNMP message as we talked about in those previous diagrams, that's how the application layer of the NTCIP model, that gets passed down to the transport layer and gets embedded into those lower layers, the transport layer and the subnet, and those protocols add on their own headers and footers depending on which standards you use. If you are communicating over an IP network, you'll be adding on something like a UDP IP header and then whatever subnet you're working over will also add on the header and/or footer.

Now if you're over copper wire as we started out with, you're going to use a much more efficient set of protocols and those headers and footers will be much shorter. The benefit, though, as we mentioned on the previous slides is the manager gets to decide when to use each object. There aren't a lot of rules in that area, which means each manager can customize and optimize its communications on the circuit.

The problem, though, is that SNMP is verbose. Every time I want to send a message, I'm identifying the full name of the objects I want to exchange. So if I start doing repetitive requests over and over again, what phase is currently green, what phase is currently green, I'm having to

send that same full verbose message every single time. And the other thing is the request and the response are nearly the same size. So I send out the message, what's the current green, I get back a response saying the current green is this. Instead of saying, shorthand, give me message number 1 and then it come back and here's just the data value. So it's not particularly efficient coding.

This results in the example message that we've just shown, two objects being 75 bytes plus the lower layers. If you do this on a 1200 bps circuit, you're looking at 1.6 seconds for those two pieces of current information, a very slow communications process. And if you're a central system trying to communicate once per second to multiple devices, that 1.6 seconds obviously is going to prevent you from doing that. There's another problem. SNMPv1 does not provide any cybersecurity protection. This kind of dates back, this was developed back in the nineties. Community name provides access control but it's unencrypted. There's no true authentication there. There's no real security. Now we're looking at saying just how to do security, particularly as we move into the connected vehicle environment. There is a movement towards that. That's being done with SNMP v3 and (D)TLS.

There's Module CSE 203 (Adding Security to NTCIP) that provides much more detail about that information. Go take a look at that module and you'll see where we're headed with NTCIP security on that. But this course will focus primarily on just the bandwidth considerations right now.

So NTCIP developed a configurable protocol in order to overcome these bandwidth limitations. That protocol is a simple transportation management protocol, STMP. It is a custom design for the transportation industry. We are the only industry that uses this and quite frankly, it's really limited to traffic signal control and signal system masters. Maybe be one or two other devices, but it's a very small niche market that supports this protocol. And so the optional enhancement of NTCIP is completely based upon SNMP and in fact, SNMP's requirement in order to support STMP, but it does compress things quite a bit if you add the support.

Let's see how this works. Using that same exact example before, now our information layer only includes the data we want to send back and forth, the actual values, not the object names. That data similar to before gets passed to the lower layer, in this case, it's the STMP layer of the application layer, and that also has a very short header at this time, a single byte that indicates that its response or dynamic object number 1. It supports a total of 13 different dynamic objects. That then gets past the lower layers. And as we mentioned, exactly what will be there will depend on your lower layers. In practice, though, STMP is typically used only with that copper wire type environment, so you're talking about very short footers, very short headers to result in a very small overhead of the packet.

Now we mentioned that there's 13 different dynamic objects. It's up to the manager to configure exactly what's contained within those objects and when to send those objects. So there are some limitations of this, but you have 13 standardized messages and you think about this is primarily designed for those very frequent messages passed back and forth, then that's more than enough to meet your needs.

So how does STMP compare to SNMP? Well, they get requests in the set responses. So when I get information, I don't need to pass my data along with it. I'm trying to get the data. When we get requests, it has empty data pass with it. And the set response, likewise, if I'm setting information I pass the data, the response is simply an acknowledgement that that packet was received. So both of those omit the data fields. The example message—that means the example message is three bytes in one direction and one byte in the other direction plus the lower layers. That means I can cover a copper-paired wire or 1200 bps the exchange only requires .16 seconds.

Now if I need to communicate with multiple traffic signalers—traffic signal controllers once per second, I can. I can handle perhaps up to six different controllers on one circuit. Now the dynamic objects themselves are configured via SNMP. That's what we meant. STMP's an extension. I still need SNMP to actually configure these dynamic objects, but that's something I only need to do once and then I let it go for perhaps years before my system changes or I need to change something. So the concept here is I could either do that in the shop before I install the controller in the field or perhaps have a maintenance technician go out and do that over a hard wire connection, direct access signal where I have a much higher data rate and I can support SNMP a lot easier. The key is that every system configures dynamic objects to meet their needs. There's a standardized way to make sure that multiple different implementations can all work together to understand these dynamic objects.

The great thing about STMP is it reduced the overhead. About a 25:1 reduction for every 1 byte piece of information you want to exchange. Now not all of our data is 1 byte; some of our data is multibit, but that name that went along with that 1 byte of information tended to be about 25 bytes long. So we reduced a huge chunk of overhead for every data element and if that data element is only 1 byte, that's a 25:1 reduction. Both of the protocols provide flexibility. I can configure my dynamic objects in STMP and I have 13 of them to configure. SNMP, of course, you have all of the data available at your availability. STMP does require full support of SNMP.

And then the downside is that STMP increases the demands on your processor, on your memory, and on your code base, which means it's much more complex. It's going to, you know, systems that are barely handling SNMP are going to struggle much more to handle STMP and the reason is because it has to translate this message from “please implement dynamic object number 1” to “oh, that really means the set of data”.

Now I have to process that data and implement my response and re-encode it. It also uses a different set in encoding rules, which once again, that means I have additional code that figures out exactly what that encoding looks like. The other real downside is this is really, once again, it's a customized protocol with a niche market that increases integration costs, increases testing costs and everything else. So there are real costs involved with STMP, but it was designed for a specific purpose and it met that purpose at that point in time. Since then, another approach has been suggested and has started being deployed in some areas. And so their approach was rather than trying to communicate to every signal on a once-per-second communications level, let's change the way we communicate.

Since my main interest is in discovering what has changed, rather than constantly polling in my device to find out what's changed, let me have the device tell me when changes occur. So now, what happens is if I'm mainly interested in my traffic signal changing phases, then I know how

many times it might change phase per cycle? So if we assume it's a 120 second cycle, at once per second polling I have 120 requests followed by 120 replies. If I do exception reporting then all I really need to do is send a message each time a signal phase changes to green. Well, that means at most if it's an 8-page signal, I'm going to get 8 reports. Well, maybe I have any time that green turns on or off is actually the way it's designed, so I have 16 different reports I have to report although that's an absolute maximum.

In practice, a lot of those changes are going to occur simultaneously, so as you cross the barrier and things, greens come up and go away. They tend to at the same time. So rather than having 16 messages back and forth for an eight phase controller, I might only have 12. So I can also delay those settings even further if I simply say I know my yellow time is 3 seconds long, so rather than having one phase say that it went off followed by 3 seconds later another phase saying it's going to turn off, I can create a delay to every time these messages occur. So then all the sudden, instead of having 16 or 12, now I'm down to 8 or maybe 6 imports every message of 120 seconds. So that results in perhaps—oh and then finally acknowledgements can also be suppressed. So rather than having a get response and/or get response, now I only have the response or the trap coming back to me.

The result is something in the range of a 20:1 reduction in communications demand, a very similar ratio of what we achieved with STMP but there's some real benefits to this approach. It could be used to capture transient events. So now things I might not actually want to pull for every second because that's taking extra data, now these really odd events like the cabinet door opening, I can create a special event for. And rather than having 13 dynamic objects that I get to choose from, I now have 64K of conditions I can define and create a different message for each one of those.

And the really good news is this logic is already being implemented in most of our controllers today. The logic to detect those evens and everything are the exact same logic that was used for creating the event log and most of our controls that support an event log. So there's only a marginal cost of identifying additional logic to implement this within the controller. Now there are some challenges in doing this. The current design of NTCIP 1103 is based on SNMPv1. We mentioned SNMPv1 does not include cybersecurity. SNMPv3 does support cybersecurity but it's going to require some changes to the current design of its exception based reporting.

The good news here is that that is well underway. There are drafts of these documents, one of which is about to be approved, so those are well under way. They should be ready fairly soon. That's the good news. There is an issue that event detection can increase processor requirements because it's constantly monitoring these conditions although there are some ways to manage that as well.

So if you want to go to a secure process, we have to recognize STMP has no security built into it other than being difficult to understand. But SNMPv3 does have full cybersecurity built into it, so that should be a consideration, part of your migration plane. Okay, so that is kind of a background of why SNMP exists. That's still the major standard we use today.

STMP was developed primarily to support lower bandwidth requirements, however that doesn't have a future path for cybersecurity as defined currently, so exception reporting has some

advantages there and that should be considered going in the future. We'll now move on to the next major topic, which is block object. We talked about the size of SNMP messages and if you're trying to download an entire database of controller information, you're probably not going to be able to do that with a dynamic object because you only have 13 of them. So how do I download an entire database with this verbose protocol? The answer is we've created these block objects.

Block objects are nothing more than really, kind of, those exact same dynamic objects except we've actually standardized the content. They're not dynamic anymore, they're specified in the standard; this is the block you get. And so that the block objects are designed specifically for the purpose of downloading and uploading the configuration for the traffic signal controller. It contains a static structure of the objects that are required to do that and it's only standardized for upload and download; it does not support currently standardized objects for status or anything else. But the manufacturers can and often do define their own block objects to manage any extensions that they take to the standard. So they can handle their upload and download, their custom data efficiently as well. The big benefit here is that it reduces its time to transfer and entire configuration. Now in practice, even then, even if you do this, you're probably going to do this in the field on a laptop or a USB stick or something, but this still reduces the time required to do this.

The next big topic to consider when we look at infrastructure issues is: how do the various controllers out in the field today and how do those communication environments affect the way that we deploy NCTIP? So across the top, you kind of see the different models of car and the point I want to make here is that when we talk about legacy mainstream in emerging controllers and field communications equipment, think about it in the same way as cars you see on the roadway. We're not talking about legacy mainstream and emergency controllers rolling off the showroom floor; we're talking about what you actually see in the field.

So the mainstream is not what's being sold today necessarily, it is primarily what's out in the field. Legacy is the minority of what's out in the field but are—still potentially create some real issues. So with that context, let's look at this. Legacy, we're looking at very low data rates of 1200 and 9600 bps. This is traditionally your twisted copper pair wire or your dial up modems; dial up might get you a little bit higher data rates, but still relatively low kilobytes per second, not megabytes per second. Processor speeds, here we're talking about your NEMA TS1 controllers, your type 170 controllers. These are very slow machines in the range of 0 to 4 bits per second. Millions of instructions per second. These controllers do not support their own operating system.

Everything is done from where? There's no ability for them to support connected vehicle applications. So if we start looking forward in what our transportation structure needs to do in the future, there is no upgrade path really for connected vehicles with this legacy equipment. It doesn't support the processing speeds. It doesn't support an operating system environment to run connected vehicle applications and it doesn't support the infrastructure—the communications infrastructure doesn't support the data rates necessary to support this information. The goal there is that needs to be replaced.

So let's look at the mainstream controllers, the bulk of what's deployed in the field today. Typically, you have some reasonable data rates, very common Ethernet or LTE, fiber gets you up in the range of 10 megabytes per second and you have reasonably modern processors. This

includes, like, the 2070 controllers and the NEMA TS 2 dated back in the, like, 2010 timeframe. And these were controllers that got you from 4-60 MIPS. Typically, they did not support an operating system with API but some of them did or some of the newer ones do. So here we're kind of in the middle area. And then we have these emerging controllers that actually most of the equipment you buy today and certainly the equipment you should be buying today are the more advanced controllers.

And when you install infrastructure you should be looking forward to these higher speeds, Ethernet, Wi-Fi and 5G. You should be looking at within this category we include everything from 60 MIPS and up; modern controllers today are closer as of 2020, are about 450-500 MIPS or faster. The current processing speed on a lot of desktop computers are over 1000 MIPS. So these controllers should also support operating systems for connected vehicle applications; an example with Actuated Traffic Controller -ATC 5201. So pretty much everything you buy today should be in that emerging field. And the question is: can we make these older controllers work?

Before we get into that, though, let's consider why we're able to do this. If you look at the legacy controllers when they were first released back in the late seventies, the cost of processing power was very expensive and even the fastest processors on the market were not that much faster than the processors that we installed in the field.

The result was our equipment was very expensive. Over time, you see the gap between what were installed in the field and what is available as the industry keeps growing, which is great because that means we can probably get our processors for less and less and that's what the orange line shows. If the constant drop in processing costs for the given speed. So the result is we can now buy an emerging controller for much less than what the legacy controller was when it was first released, at least that's the theory, right.

In practice, what happens is you end up with some market space and agencies are willing to spend so much on a traffic signal controller. So what's happened is in the late seventies, everyone was electrical-mechanical and they were hesitant to even buy these new microprocessor-based controllers. But as their price dropped then they became more reasonable to deploy and most agencies were willing to deploy them for about that $2,000 to $6,000 sort of price range. And as they dropped further to that $2,000, everyone started deploying these in the eighties, right.

And then as they got deployed, then we improved signal controllers and everything else but the logic, natural signal control logic on these processors. And for a long time, we said, “well, why would I even need a faster processor because the existing processors do what I want?” But over time, there's a demand for better user interfaces, better logic within the controller, more advanced features and functionality. And what you saw was kind of this desire. Initially, the prices were at that $6,000 range, dropped down to the $2,000 range and then new features crept in and then they started—prices started going back up to get the nicer features and the prices have kind of stayed in that range.

But over time, we've seen processor speeds increase. We've seen memory increases. We've seen support improve, support faster communication speeds. And every new version, results in reengineering the systems. So but the prices have kind of stayed in this range of $2,000-$6,000

in current day dollars of 2020 dollars. And over time, now what we're seeing is within that price range, you can buy emerging controllers. And those emerging controllers can support connected vehicle applications, which is where we need to be to support the coming vehicles within the next few years. So now that price range also varies on a number of factors. It varies on what features I want, okay, how much processor speed do I want? And what sort of proprietary features do I want? What sort of support do I want after I purchase my equipment?

That also depends on the type of software I get. You know, am I getting just basic, so, controller software? Am I getting some sort of advanced algorithms? What sort of purchase quantity? Is this a state-wide contract or is this a contract for one particular construction contract? And then finally, what sort of testing requirements am I going to impose on this? All of this is going to affect your controller costs which is one of the reasons why we have such a wide range of $2,000-$6,000 for essentially the same signal controller. That's roughly where we are in the marketplace. So the prices have come down for the processor itself and for all these technologies, but every single cycle there's reengineering and everything else and the reason you see that price be in that ballpark range is because of all the engineering costs associated with that.

So let's look at the infrastructure limitations for each one of these devices. So if we look back at our older car or older signal controller, what challenges do we have? Well, on the communications network side, 1200 bps, it's going to require something more efficient than simple network management protocol. It's going to require either STMP or exception reporting. There's going to be no support for any cybersecurity. That's going to be due to insufficient bandwidth because we're talking about communications right now. And it should never be used with connected vehicles because if you don't have a secure system, your security of your entire system is only as secure as your weakest length.

So if this length cannot support cybersecurity, you should not be deploying this in a connected vehicle network. Now legacy controllers as well, they also struggle to support NTCIP because their processors and memory are going to be too limited to support everything that's needed for connected vehicles. There's limited support with later model upgraded legacy controllers for NTCIP without the connected vehicle environment. So just basic NTCIP operations, you can get limited support for but the original processors of these controllers were too limited.

So if you look at the original 170, for example, it requires a processor now on the extension module or somewhere else or an upgraded 170 design to actually support NTCIP. The original controllers, once again, they're struggling to support basic NTCIP, they're not going to be able to handle the processing load for cybersecurity and all of the algorithms required there. And then finally, they're not going to be able to support your connected vehicle applications. So the bottom line is looking forward, we need support to connect the vehicles; that means we need to upgrade these controllers very shortly to emerging style controllers as well as your communications network.

Let's now look at the mainstream systems. This, once again, is not mainstream of what's rolling off shop room floors; these are the controllers that are filling our controller cabinets today in operation. As far as communications networks are considered, the 10 megabits per second is more than enough to support SNMP for even reasonably large sort of controller networks. You'd have to do your own data analysis to know your own needs, but generally speaking, it's not an

issue. Cybersecurity, though, does become rather critical because if someone accesses that network at those high data rates, that probably means that they have access to a very large number of traffic signal controllers, so security concerns have risen quite dramatically.

Once again, see Module CSE 203 for more details about how we're looking at adding cybersecurity to NTCIP as well as what you can do in the near term. As far as the controllers themselves, Ethernet communications can overwhelm some of the earlier mainstream processors. So what they discovered was they took some just kind of processors of the day, added on the Ethernet card and all of a sudden they realized once that Ethernet channel became really busy, that that processor would start getting burdened by those communications.

So obviously, the way to avoid that is by not allowing your internet channel to become that busy. There are some ways to prevent that at the local traffic signal. You can also upgrade the processor or redesign the controller to handle that. Most any new system you get today should solve that problem but be aware some of the older controllers might have that problem. Is support possible? For most of NTCIP functionality, pretty much everything that's in the standard except some of the higher end, some of the tables can grow quite large and depending on how much memory you have, you might hit some limitations there, but all of the base functionality you can easily support.

You might bump against processing memory limitations as you go along for very large logs or event reporting that are complex, things like that. Cybersecurity will introduce additional processor loads, so that should be of interest. That probably starts knocking out some lower end mainstream systems, but the more modern mainstream systems should be fine. We'll have to look at each one exactly what it can support. So the connected vehicle application might stress some of the systems but you'll need to do testing and figure out what can be supported. And many of the systems, they're perhaps new enough that you might be able to upgrade them. We certainly were able to upgrade 170s for a very, very long time and make those work.

So you'll have to look at your own system to determine details, but there's a reasonable amount of good news there that—at least at a minimal level—you could probably support some connected vehicle applications. So the recommendations is we should continue to migrate to emerging controllers as needed—that's kind of a case-by-case basis—and initiate a migration plan to provide cybersecurity. For the emerging controllers and communications networks, they support NTCIP. You have more than enough speed, network loading investment generally is not going to be an issue; however, once again, particularly as we move into wireless type environments, cybersecurity becomes a truly critical issue that is really important at this point.

Emerging controllers, they can also handle requirements. Most of the controllers you buy today should be looking at 450-500 plus (+) MIPS. Once again, though, we're now getting into modern processors. We're getting into processors that hackers might actually be interested in processing. They're running LINUX, a well-known operating system for the hacker community, so not only are you increasing access through your communication system, you're increasing the value of the target and now you're potentially introducing this in the connected vehicle infrastructure, also increasing the value of the target and the safety criticality of that target.

So I can't overemphasize enough that cybersecurity is critical for this environment. We should absolutely be initiating migration plans to provide cybersecurity and buy more than enough processing power when you buy your new equipment.

So the good news is this chart was developed in 2012 as a part of the development of this course. It is designed to project what the controller market would look like over time with a couple of data points from 2011 and 2012, we saw at that time even most new signal controllers were replacing legacy equipment with newer equipment. So the chart shows the red line is legacy equipment. The blue line is mainstream emergency equipment. The green line up at top is just the total of the two. And you can see in 2012, we were projecting that by roughly 2018, we should be able to eliminate pretty much all of our legacy equipment.

At the bottom, you see some actual statistics from 2019. Now the upper chart was based on some studies of the entire U.S. marketplace. The lower is just for Minnesota. But presumably, this is reflective of the country as a whole, that legacy equipment is now a very small market share of the installed base and obviously shrinking over time. Mainstream is in fact mainstream now. It's 65 percent, two-thirds of the market space, and emerging is 35 percent. And I just noticed that adds up to more than 100 percent, but suffice to say, that legacy is a very small percentage, 5 percent or maybe less, and we are emerging—moving towards an emerging industry.

Communications demand versus capacity. So that kind of completes the one thought of how are different controllers working. Now let's look for any particular deployment, how does my communications environment work. When I go to deploy my communications and design my communications system, I need to be looking at a few things. I need to consider what my communications loading is going to be on my network. That's critically important if you have a legacy system. In practice, though, hopefully you have a more modern system. If I limit my capacity too much, if I have a legacy system I really constrain myself with capabilities but recognize, if I open up my system too much, I'm potentially opening myself up to traffic from hackers or something else that might be difficult to notice because I have so much space it's not impacting my operations right now.

That means as I open up all of this bandwidth, I also need to be taking cybersecurity more critically. So determine how much data per second from all devices is required to share the communications medium. Particularly on Ethernet you're going to have multiple signal controllers all sharing a common channel. At what point do I start overloading that system? Well, once you actually start doing the—Oh. And then you also need to make sure that for supplying Ethernet, you need to make sure that you allow kind of double your estimate in communications loading to allow for collision detection and peaks in demand and other things. But once you do that analysis, you start realizing fairly quickly the data to talk to traffic signal controllers is relatively low compared to your video demands.

Once you add a video camera to your system, you have an interest in receiving live video from an intersection, then that's going to dwarf your data demands. In fact, video requires something in the range of 1.54 megabytes per second. That's 1500-4000 kilobytes per second. By comparison, SNMP, that very verbose protocol that we're talking about, only requires 1.54kbps per signal. Meaning I can have a thousand signals passing SNMP data around is roughly equivalent to a single video that's worse. So even if I had 100 signals that video is still going to

dominate 9:1 of my bandwidth performance. So generally speaking today when you're looking at your communications analysis, it's more of a video analysis than it is a data analysis. But you do need to be aware of that in your communications design.

If you do have legacy communications and you're not passing video, then you have to consider what your demands truly are. But let's take a look at what that legacy data means, that legacy infrastructure. Legacy communications means you don't have cybersecurity because you don't have the extra bandwidth to throw on signatures and everything else you need to provide that cybersecurity. It also means you've increased your equipment costs due to custom code to support these very efficient protocols. It also means you've increased your testing and integration costs to make sure that they've implemented those protocols correctly. And you have to deal with the fact that you're going to see fading industry support in the future because we just saw these legacy equipment is fading away from the industry. So increasingly, you're dealing with a smaller and smaller and smaller niche environment, you're going to have to support all of the engineering costs to develop all of this and maintain all of this yourself.

So if you have a legacy communications system, consider upgrading. I mean, wireless systems are often viable and if you put proper cybersecurity and other techniques in there, they can be secure. And also use a touch base reporting rather than STMP because that will be a protocol that is supported for a while to come, for a long term to come. View STMP as truly a last resort because of the cost involved in it.

So with that, we're up to our first activity. The question is: which of the below is a waning technology that is not recommended for most new deployments? Your answer choices are: exception reporting; block objects; STMP; or none of the above. So go ahead and make your choice and we'll review the answers here in a second.

Okay. So let's look at the answers. The correct answer is STMP. STMPs are protocol-specific to the transportation industry meaning it's very customized to our industry. It requires extra testing and integration experiences. As a result, it's waning—it's falling out of favor, therefore becoming more and more and more customized, meaning you have more and more costs involved with it and then it becomes a downward spiral. Exception reporting is a fairly new feature that has been added to NTCIP to reduce overhead in communications. Block objects are used to speed the upload and download of large portions of the ASC database. And obviously, if C is correct, then D is incorrect. STMP is a waning technology.

So that completes our infrastructure review. Now let's look at our functionality. So with our functionality we're going to be talking some about database transaction sets. We'll talk some about consistency checks and rules and connected vehicle supports. So the first two talks largely about how download operations work and how reconfigurations of major portions of controller work. Then we'll get into connected vehicles with support. Also, clock coordination. It's very tightly related to the connected vehicle discussion. And then we'll talk some about managing your expectations for truly off the shelf interoperability.

So with this, let's consider the need for complex transactions. Traffic signals clearly are safety critical devices. They're controlling access to an intersection. It's very important that our databases do not become corrupt in the field otherwise, the good news is we do have

infrastructure in the cabinet that prevents conflicting signal displays or just go into flash, but we obviously don't want that to happen. So how do we make sure that when we download a new time plan we download a new configuration to the device that something isn't going to be wrong which causes a problem? Well, many of our configuration parameters are interrelated and changing the configuration of some parameters cannot happen independently of other parameters. They have to all be implemented in a single step.

And we have a challenge that our SNMP message that we talked about before, the size of that message is limited. It can only grow so large and then it says well, no, I can't handle this message. So how do we handle both of these competing conditions? I mean, to be able to download a large hunk of data but it all has to happen at once and I'm limited to how much I can actually send down. Well, that's the purpose of the transaction mode of NTCIP. It allows us to interpret multiple SNMP messages as one transaction. The way it does this is it takes some individual messages. Once you're in that transaction mode then it buffers these messages and it stores those requests without actually implementing them until you actually then tell the transaction mode to go into the next state and actually implement all of those messages.

Let's look and see how this works. When in transaction mode, okay, so put the device into the special mode, some operations are buffered. Now not all operations are buffered. I can still do a get request and get live status variables from the device, right. I can do set requests for control objects. So I can tell my controller to go ahead and switch to pattern number 3 and it will go ahead and do that because that's a control command, not a configuration command, while I'm in this transaction mode. But my data, my set requests for database parameters are buffered. So if I want to reset the minimum green time, that's the type of command that would be buffered. And some parameters like phase concurrency, the ability for phase 1 to time with either 5 or 6 that is a condition that can only be changed in transaction mode because of the complex interactions that has with everything else.

If I want to change phase 1 to only tying with phase 2, then that's a completely different design and I don't want to implement that one change by itself. I would get myself into problems real quick. Within the standard in a very broad term, each of these objects are defined as which parameter type they fall into. However, the section was accidentally omitted from the version 3 of the standard. They are in version 2, okay. So for all of the objects that were defined in version 2 there is a standardized definition of how this database transaction mode works. We did add a fair number of new objects in version 3. There is not a fully vetted version of exactly how to handle all of those, although in practice most of them are very, very obvious. But right now, we're still waiting for an update. The NTCIP workgroup has been made aware of this problem.

So once I go into transaction mode, remember, the reason that this mode is required is because I have some objects that require this type of operation. The reason they require this operation is because I have consistency checks I need to perform. If I say signal phase 1 in time with phase 5, but phase 5 is not defined at the time of phase 1, then I have a logical inconsistency in my database. So I have—there's the standard defines it—17 specific checks that must be performed when exiting this transaction mode. So I have a set of critical parameters that require you to use this mode, and as I exit that mode and implement those new settings, the controller will go through 17 consistency checks, and it prevents implementation of any internal conflicts of those settings.

It requires additional time, so it's not an instantaneous check. It does require some time, particularly on older controllers, and they're performed at the end of the process, not on every request. The manufacturer can impose additional consistency checks. This is to protect them as well as you. They may discover that there's some other check that should be performed and they can implement that immediately without having to wait for the standard to catch up. Failure of any rule results in a transaction set failing, and then you have to kind of start over and re-download what you want.

So what are these consistency checks? Well, we're not actually going to go through all 17, but we do want to give you a flavor of what's been done. So if I look at my standard NEMA diagram, eight-phase signal controller with all my different directions, we have two rings: Ring 1 and Ring 2. Ring 1—traditional NEMA diagram is Ring 1—consists of phases 1 through 4. Ring 2 consists of phases 5 through 8. In the middle, and actually at either end as well, there is a barrier. That barrier separates Conformance Group A from Conformance Group B. Or sorry, not conformance—Concurrency Group A to Concurrency Group B.

The logic of the diagram is that Phase 1 can time with either Phase 5 or 6. Phase 1 can be concurrent with any phase of the other rings that might exist. So the standard eight-phase diagram, you have two rings, two concurrency groups, but the standard in NTCIP allows a very large number of concurrency groups and allows a very large number of rings. Most intersections only require this sort of design though. So I could define a 16-phase controller with four concurrency groups and three rings, and all sorts of strange combinations, but this is the standard signal design. But because the standard provides all that flexibility, then when I configure my controller, I need to be very precise about how everything's supposed to operate.

So there's a table that tells the controller what ring—one parameter says, "What ring is Phase 1 in? What ring is Phase 2 in? What ring is Phase 3 in?" What you see here is the Ring column, or Ring row, indicates that Phases 1 through 4 are all in Ring 1, and that Phases 5 through 8 are all in Ring 2. But to know how the signal times, I also need to know other details. I need to know can Phase 1 time with Phase 8. The answer is no; that's on a different-sized barrier in my diagram. The controller doesn't know about the barrier, per se; it only knows about concurrency groups.

So what we define is that Phase 1 can time with Phase 5 and 6. So Phase 1 times to Phase 5 and 6. Phase 2 could also time with Phase 5 and 6. Phase 3 times with 7 and 8, and so on. So the first rule, or the first consistency check that we've defined, sign that the concurrent phases must be in different rings. I can never assign Phase 1 to time concurrently with Phase 2. They're on the same ring. That is prohibited. And if I configure my device with a 2, this concurrency, then at the end of that consistency check, at the end of my database transaction mode, I will get back an error and it will prevent me from actually implementing that timing plan.

So that's what that consistency check is designed to do. There's another check to make sure that if Phase 1 is timed to be concurrent with Phase 5, then Phase 5 also has to concurrent with Phase 1, and there's that parallel there. So there's all of these 17 different checks that are performed to make sure that what's configured in the device is a safe configuration, and a logical configuration.

So that's what Database Transaction Mode is. We'll now move on to a discussion of some of the connected vehicle technologies. So let's look at what version 3 of NTCIP 1202 does now. This is kind of an adaption of the diagrams used by ARC-IT, the Architecture Reference for Cooperative and Intelligent Transport Systems, and that's the major reference architecture for ITS, and what you see here is in teal you have a traffic management center. That traffic management center is talking to both your ASC, what that architecture calls ITS roadway equipment, and your RSE, what the architecture calls connected vehicle roadside equipment.

Now, those two boxes—you see these dashed gold lines—those two boxes are both in that gold color, and the dashed lines indicate that that may actually be a single box or it could be two separate boxes in the field. I might have my RSE separate from my ASC, or my ASC and RSU could be inside of one physical box. NTCIP 1202 defines communication for the TMC down to the ASC, as well as the TMC down to the RSE for some specific signal-related communications. It also defines communications between the ASC and the RSE.

So 1202 covers all over those data exchanges. The RSU, its main purpose for existing is to communicate with the vehicles and pedestrians and other entities outside through this DSRC-type connection, local short-range wireless type connection to vehicles, given them signal timing information and things like that, and that connection is a different standard. That's 2735 of SAE, SAE J2735. But that data, bidirectional— so data is being received from vehicles and it's sending data to vehicles—and then likewise that roadside equipment is exchanging that information with the ASC and with the traffic management center. You'll see this other number here as well.

NTCIP 1218 is another standard related to the roadside equipment. That covers more kind of fundamental data about the RSE itself, unrelated to traffic signal information. So if you're implementing that RSE, you need both the traffic signal information of 1202 as well as the 1218 to support a signal application inside of your RSE, and that's what this box is here, RSE intersection management, is that additional information that's supported by NTCIP 1202. We'll talk a little bit more about that in a couple of more slides. The other white box you see is roadway signal control. That will also appear in a couple slides from now.

But let's consider exactly what data was added to NTCIP 1202 to support the connected vehicle environment. For that TMC to ASC exchange—so directly from the management center down to the traffic signal—we manage data exchange between the ASC and RSE. In other words, the traffic management center needs to be able to configure the ASC to tell it what data it can give to the RSE. That's its full functionality, is kind of a management level, since that management—you don't want to have to send out the technicians to field device direct. You want baseline configure that from central. From the TMC to RSE, you get a little bit more information. Not only do I have to be able to manage that remote device, but there's some information I want to send that device directly, in particular the map information.

That map information is how the RSE communicates to the vehicle and says, "Here are the lanes approaching my intersection, and here are the movements allowed for each lane on the approach." That's how a vehicle needs the intersection to know, "I'm in this lane. I can make this movement from this lane." It then translates that and says, "This movement from this lane relates to this particular phase of the signal controller." So it's the RSE that manages this transition between the phase logic of a traffic signal and the movement logic of vehicles. The

RSE sits in the middle of those two images of the intersection and translates from one to the other. It takes Phase 1 and it says, "Oh, Phase 1 actually means this particular approach on this particular map of my intersection." I then say that this map has a green for this movement, as opposed to this phase.

The vehicle doesn't know anything about phases. So it's the RSE that makes the translation between movements' maps and phase information. That RSE also manages the data collection from the basic safety message, or the personal safety message from pedestrians—it manages that information and then can pass that up to the TMC. The TMC is able to configure what data do I want from those BSMs and PSMs, and then the RSE collects that data and then passes that data up, and all of that is defined as new objects within NTCIP 1202. The third category of information in this standard is the interface between the ASC and the RSE, those two field devices. The ASC needs to provide current next-movement information, as far as the phase information. Then likewise, the expected start and end times of each phase.

It also provides—so actually the movement information is, "When am I going to change my timing plans?" So if I have my time-of-day operation of left turns being prohibited using peak hour, then it's the ASC that needs to tell the RSE, "I am now shifting from my midday timing plan to my evening timing plan, and my left turn on this phase is no longer allowed." And then that can be reconfigured—the RSE can then know, "Oh, I'm now changing the maps of what movements are allowed, and I now have to convey this map information out to my vehicles." So that's the current geometry and the current transformation of ASC timing data to SPaT message, so which set rules apply for that mapping.

Then RSU is able to report the presence of vehicles and vulnerable road users—so pedestrians, for example. So the RSU can be used, probably not so much today because you still have to have traditional detection to detect non-connected vehicles, but in the future, you may be able to start relying much more heavily on the BSMs detected by the RSU, and then pass to the ASC so you can better time your traffic signal based on current demands. That's also within NTCIP 1202, the new version.

So, what are the challenges to doing this? Clearly, connected vehicle environment is safety critical. The integrity of data must be maintained. Absolutely, without a doubt, if the RSE is going to be telling the vehicles, "This movement is now green for you to go over," the integrity of that data and the accuracy of that data has to be absolutely precise. That means your messages must be properly authenticated, not just from the RSE to the vehicle, but also from the ASC controller to the RSE itself. And the access to that data must be controlled. I need to make sure that there's not someone else on that communication circuit that has hacked in and is trying to inject information to my RSE that would corrupt that information.

So the roadside unit specification, because that RSU specification was designed specifically for this purpose, it made sure that we only allowed secure protocols. The only version of SNMP that was allowed by that specification was SNMPv3, but 1202 version 3 was designed for SNMP version 1, which is not secure. This creates a problem. This creates a challenge of, "How do I get my ASC controller that has data for my RSE, how do I get that data securely into my RSE?" The short answer is you should really be upgrading that to SNMPv3. We'll discuss the process for that in that other module I mentioned before, CSC 203.

In addition, as of 2020, that Security and Credentials Management System, SCMS, is still being established. It is sort of kind of getting there. A prototype was established for the pilot deployments. That is in kind of a temporary operations state; they're working out details about how that will be maintained long-term. They're also finalizing some of the standards so that we can scale this system out much broader. Right now, there's a lot of manual configuration that has to be performed, but that will change once this next standard gets approved from IEEE. So all of that's underway, but there is still, as of 2020, some challenges in doing this full-scale operation.

In the meantime that we don't have truly secure communications, until you get to that point and everything's been tested out, the data being sent from RSEs and SPaT messages should be considered informational, not normative. So if there's a conflict between what I'm getting from SPaT and what I'm actually seeing on the signal head, it is what is on the signal head has to take precedence, and we don't have full confidence as of yet in what's being broadcast from traffic signals. That doesn't mean you don't deploy it today; it means that you need to make very explicit that what's being transmitted is for experimental use at this point, but that experimental use is assisting not only the different systems manufacturers to make sure we've worked out all the bugs, but also we're not essentially in the stage of deploying that, so that actual operators in the field get experience in deploying this equipment and understanding how to use it, configures it, and everything else.

That's where we are today, is that it's ready for deployment as experimental use, and as of 2020, within the next few years, hopefully we'll be able to actually go live with true use of this data as we move to a more secure system and get the SCMS established. So the connected vehicle features of NTCIP 1202 version 3 at this point in 2020 should only ever be used in a fully secure environment. At this point in 2020, we're not there yet. That does mean we need to move to a secure communications, which meant SNMPv3 over TLS, and we have to have proper maintenance of SCMS certificates, which is also currently being developed.

So another issue that we face when we start sending out information from a signal controller to a connected vehicle is how do I synchronize all of my time sources, so that when the signal says it's going to change in three seconds that my vehicle also understands that three seconds to be exactly the same point and instant within a very, very tight margin? Well, let's consider first where we've migrated from and where we need to get to. So traffic signals need to consolidate timing with adjacent signals and connected vehicles. The first has been there for a long time, so let's take a look at it. For inter-signal coordination, we need precision within a couple of seconds because the point of your precision is to make sure that my signals arriving from the upstream signal get a green as they approach the intersection, and then they can proceed smoothly in order to have proper signal coordination.

But the drivers can adapt a little bit so as long as the signals are timed consistent, within a couple seconds of each other, that's enough for signal coordination. And my accuracy really doesn't need to be all that precise. As long as it's within, oh, maybe five minutes, then my timing change from my overnight plan to my morning plan, my morning plan to my midday plan, and my midday plan to my evening plan, those timing changes are—they need to start on time, but if they're off by a minute or two or even five minutes, it's not the end of the world.

Then you start going to 15 minutes or more, you start getting into problems. But let's say accuracy within about 5 minutes is what I need. So the chart at the bottom kind of shows what we mean by that. So if 8 a.m. is when I schedule everything to occur, and I have four different traffic signals over here at roughly 30 seconds later, and that's when they all think it's 8 a.m., but they're within one or two seconds of each other, then I will have coordination along my arterial. That will be fine.

If my signals change or timing plan is 30 seconds late, it's not really going to be a huge detriment or anything. That's enough for signal coordination. That's not going to be enough for connected vehicles. If the connected vehicle thinks my yellow is going to come up 30 seconds later than it actually does, I have not achieved anything, right? I need to be much more precise.

So that's what this chart shows, is everything is kind of backed up on itself. The coordination with connected vehicles requires 50-millisecond precision. So if our yellow times are measured in one-tenth of a second, or accuracy here, or our precision here, has to be roughly half of that. Likewise, our accuracy also needs to be there as well. We also [need] a single reliable national standard to synchronize. Previously, with traffic signals, I could have a signal system on my west side of town completely different time source than the signal system on my east side of town.

So now, all of the sudden we have vehicles that are traveling all across the country and all of those vehicles need to work in synchronous with my local traffic signal, which means we need a single source for time. Luckily the computer industry solved this problem. They've done [this] in a few ways. They've done it through network time protocol—works great for wired devices or connected devices back to central. In our environment, it's probably much better to use GNSS time, Global Navigation Satellite System, also known as GPS, Global Positioning System. GPS technically is the U.S. system. That's the original U.S. military system. There are other systems deployed today. So the generic term is GNSS. So even if I travel outside of the U.S., if I'm traveling somewhere, it's still a GNSS time. So assuming satellites are accurate, then I have accuracy and precision at the same level and it gets me to that 50-millisecond level of precision, which means I can have all of my signals and my cars all stacked up right at 8 a.m., and we all know exactly what time it is.

Now the interesting thing is maybe in the interim I have a combination of designs. I have some traffic signals that are not yet connected, and then I have some newer traffic signals that are connected. So now, I can't necessarily say that all of my traffic signals have and use GNSS time. I have to coordinate my signals off of one time source, but then I have to communicate that time to my vehicles based on GNSS. So how does this work? How do we translate this information? This is all defined within NTCIP, and the basic way it works is we talk about that roadway signal control functionality within that box in the field. That can be based on any time source, and it might be minutes off of GNSS, but it should stay within a couple of seconds of upstream and downstream signals.

That's its goal, is to coordinate signals amongst each other. When that roadway signal control, my ASC box, communicates with my RSE box, it's going to send this flow called intersection control status. That's a flow from the national architecture, from ARC-IT that we mentioned before. For example, it's going to say that, "Right now, according to my roadway signal control, it believes it's exactly 8 a.m., and that's when my cycle's starting, and I'm going to change now

in 4.3 seconds," and that's the information it sends over, that, "I think it's at the top of the hour exactly, 0.0 seconds, and I'm going to change my phase in 4.3 seconds."

The RSE receives that information and it converts it to GNSS time. So it doesn't really care about what the roadway signal controller thinks what time it is; it's really only looking at, "What is this difference, and how do I match that up, and is it close enough for me to think that this is valid information?" So it converts that to GNSS time. So for example, when it receives its information, the GNSS time is exactly 8 o'clock and one second, then it knows the time it will change is 4.3 seconds from that point, which will be 8 o'clock and 5.3 seconds. So that's the information that is sent to the vehicle. In this different flow intersection status, it will send out the time of 8 o'clock and 5.3 seconds. That vehicle will also understand GNSS time and will be perfectly in sync, within 50 milliseconds, and will therefore properly understand that 8 o'clock and 5.3 seconds, which means they all arrive at the right time and everything works.

So that completes our connected vehicle tangent, and we'll move on to the managing expectations for off-the-shelf interoperability. Signal controllers have many more configurable parameters than other field devices. It is a very complex, interrelated set of control parameters. Many agencies continue though require specialized functionality, and those specialized functions are met through sometimes agency-specific extensions, sometimes proprietary extensions, or lesser known public extensions even, developed by maybe a university. So those specialized functionalities get added to the base standard, and some details even within the standard are manufacturer-specific.

So for example, we have the ability right now to change timing patterns within the signal controller. So as I start my morning peak, I change my timing pattern and now I implement all of these new timings. As the signal controller changes to the new timings, it has to shift the way it operates, and there's different ways you can configure which way you want it to change its timing pattern. You can say that, "I'm going to short way it," which means I'm going to take the shortest possible route, either adding time or taking time away from phases in order to implement the new timing offsets and splits and everything, or I can add-only or subtract-only, which means I will only add time to my phases or only subtract time to my phases. But there's some caveats here. Manufacturers are not required to support all three modes. That's the first challenge. Even if you manage to specify that your equipment will support the mode you want, manufacturers sometimes impose restrictions on how much time can be added to a particular phase before switching.

So I might be add-only, but as I add time to my phase, one manufacturer says, "I can only add 5 seconds per phase," the other manufacturer says, "I can add 10 seconds per phase," and the result is my timing systems don't all change dynamically and interoperate together. And actually, even if you did standardize all of that, then you might have—if you select short way, one controller is adding time, another controller is taking time away—and during the transition period, you're still losing your coordination along your street.

This is a well-known problem among the traffic engineering and designing of these systems, but it is a problem. Don't expect it to go away. There are other challenges like this with other manufacturer-customized extensions that might be different from one manufacturer to the other. The NTCIP was designed to improve interoperability and interchangeability. It's not necessarily a 100 percent fix. Making it a 100 percent fix ends up decreasing the innovation capabilities of

manufacturers and then we stagnate as an industry. So we've kind of drawn a balance here. The goal is the NTCIP v03 is a step in the right direction. Version 03 has added a lot of content to solve a lot of the previous problems. It's also added a lot of connected vehicle applications. We are really moving in the right direction at this point.

So with that, we come to our second question. Which of the following most accurately expresses state of connected vehicle support in the standard? It does not address any CV functionality; it does not define sufficient security for ASCs in a CV environment; it defines a secure solution for intersection maps; defines a secure solution for signal timing. Go ahead and make your selection and we'll take a look at the answers here in a second.

The correct [answer] is NTCIP 1202 v03 assumes SNMPv1, which is not secure. For CV operation, NTCIP 1202v03 must only be deployed over a secure protocol such as SNMPv3 with TLS. So its security is not sufficient for ASCs in a CV environment, at least not as a normative requirement. As an experimental type of operation, feel free to do that. Just make sure that everyone knows it's an experimental state and not fully operational yet. The other answers—1202 v03 does provide data to support CV applications, including map and signal timing information. It's just not yet in a secure format. It does not define a secure solution for intersection maps because it's not secured, and likewise it does not secure signal timing, because it's not secured.

That brings us to our third learning objective, incorporating requirements not supported by standardized objects. Now, it should be noted, we don't necessarily recommend doing this, but we recognize it's a need of the industry, and it's how innovation grows and how we add things to the standard eventually. So we're going to give a couple of examples here. We're going to be looking at the conditions and context for when you might want to extend the standard. We'll give an example of Dilemma Zone Protection, and then we'll also look at specifying requirements not covered by the standard and how you do that through extensions.

So what is a custom extension? The standard defines both mandatory user needs and optional user needs, and for each one of those user needs, it identifies mandatory, optional, and conditional requirements. So in other words, this is a requirement that is mandatory for this optional user need, or it might be this requirement is conditional based on if you supported this user need, and if you've supported some other requirement. The standard also defines the mandatory dialogs and objects for each data exchange requirement. So every time you have a data exchange, that will trace to precisely one dialog and one or more objects, all of which much be supported.

So it's a single design of how you implement that particular requirement, and that's how we get interoperability. The customer extensions can define objects and/or dialogs—really "and" dialogs—for user needs and requirements that are not addressed by the standard. You should never be defining objects or dialogs or requirements that are also addressed. You should really only be defining objects for those that are not addressed. We'll talk about it in a second—there actually are some cases where you might want to talk about some dialogs. Actually, I think we did this in the previous dialog, some customized dialogs for the same service, to be a little bit more efficient. But objects you would never define for something that's already defined. Extensions are allowed for NTCIP explicitly to allow for innovation and growth of new features.

So why might you want to do that? Well, the standard doesn't define every single traffic control feature. It defines only those features that are known to be in wide use. If something is deployed in one city somewhere, we're not going to necessarily standardize it. The customization allows for market innovations. That's great. It allows our industry to grow and enhance and get better, and those extensions may eventually be incorporated into the standard if the industry as a whole thinks that they're valuable input.

An example of that is Purdue University developed a high-resolution data logger. This was intended for performance monitoring of traffic signals and is now becoming quite common. It's been implemented with multiple manufacturers and was added to the draft of NTCIP in 2015, and finally approved in 2016. So that's a prime example of how industry can come up with a new idea and it eventually gets incorporated into the standard, makes the standards even better.

And, while at the same time, not forcing manufacturers to wait, implement new features until it's fully standardized. Now, there are costs associated with the extensions. One is the custom features might be proprietary. Well, the whole purpose of NTCIP is to get us away from proprietary protocols into something more flexible. So by implementing a proprietary feature and your system becoming dependent on the operation of that feature, now you've kind of started tying your system into that one manufacturer. So that might be something you want to avoid. The other challenge is NTCIP standards—public information, well-documented information—if it's proprietary, their documentation might be limited and/or cryptic.

So just because they've said they will provide you documentation, telling you that this is a four-byte field doesn't really tell you that much. They need to explain it a bit better than that. That would be rather cryptic. You also need to distribute documentation to those who need it, and unless you ask for that, they can actually limit your right to distribute that information. So as the agency, you might get access to their documentation but then you only get access by signing a nondisclosure agreement, and unless you're actually developing in-house the software to interface with this, it's not going to be of much use.

So you need to make sure that if you're using something like this, you get the proper rights to distribute the right documentation to the right people. Custom features might also reduce the number of bidders for a device. So even if I develop my own application as an agency and then issue a specification to purchase a controller like this, some of the manufacturers may not be interested in implementing a new feature even if I tell them exactly what it is up front. That's new development cost, and it's only going to be for one client for one traffic signal controller, it's either going to be really expensive, or they may just no bid. And that's the other challenge, is it increases cost not only for the controller you're buying but also for your management system.

Both sides have to implement that custom data in order to make use of that custom data. Custom features, of course, complicate testing and maintenance. Developing and implementing custom test procedures is quite expensive. Documentation might not reflect an as-built if it's not tested. So they might deliver me a product, they might send me documentation, and then years later when someone else tries to go integrate with it, they realize, "Oh, this documentation doesn't agree with what they implemented," and believe me, that does happen quite a bit.

Custom features complicate maintenance as well, potentially limited to one vendor, model, and version. That means I'm having to develop—any time I want to fix this, I'm likely going to have to go back to that one vendor to get it fixed, and if it's only potentially even in one model of their controller that might be discontinued, I have even bigger problems. So there are real challenges with custom experiments. This is why we want to kind of shy you away from them. But if you have a true need for something that's not currently in the standard, there is this capability.

So let's think of an example of a potential user need that's not in the current standard that might be needed in the future. We have these new connected vehicles coming onto our streets. How are they going to work? What advantages can we get from these connected vehicles? Well, one option is those connected vehicles are sending out Basic Safety Messages, BSMs, ten times a second generally, and they're being broadcast for about 300 meters, and that's the broadcast range of those messages. Well, that gives us about a 19-second advanced detection on a 35-mile-an-hour roadway. That's a huge, huge benefit from where we are right now.

That means I can potentially have my RSE with more advanced controller logic be able to determine and track every vehicle on the approach and identify a dilemma zone for each and every vehicle and determine whether or not that vehicle might be within its dilemma zone right now. So I can develop some sort of logic here to track that path, track that information. Not only the path, actually, but its acceleration and current speed and driving characteristics, headway among the previous vehicle and everything, so when you try to characterize how aggressive this vehicle might be driving, whether it just pulled out of a driveway or just is slowing down to turn into a driveway, we process all of this information well in advance of my traffic signal changes phases, and then to optimize when to gap out, it will minimize any potential conflicts.

So let's consider this. If that's my user need, I would then have to go on to the next step, develop a set of requirements. So a couple of sample requirements might be listed here. The intent is not to get this perfect at all; it's just to give you an example of what these requirements should look like, that within 0.1 seconds of its receipt—so I've constrained its timing characteristics and everything—the RSU shall forward the following data to the signal controller for each BSM received that reports a vehicle on one of the approaches of the intersection.

Then I list out various pieces of data, and then upon request from a manager, the RSU shall be able to enable or disable BSM recording. So these are two requirements—there would be many more, of course, to make this actually happen—but two sample requirements. What we discovered here is that the list of data listed here, virtually all of that comes from the Basic Safety Message itself.

So some of that data, that data that I listed out, already exists. Some of the data will need to be transformed. In particular, the BSM data will need to be mapped to the SNMP format. So while we have the basic data elements, we need to document them in a different format. And then some data might also be new. So in this case, toggling your reporting of the BSM data will be able to enable or disable this logic; we needed a new object that doesn't exist today to do that.

Now, how do I go about doing this? We could design the whole thing ourselves, but more likely, we'll take some other approach. One approach would be to say, "You know what? I'm not the expert in the requirements. I'm not the expert in design. I'm going to put together what my user

needs are, because I am a user, state them as much as I can tell them, put that in a procurement, and let the vendors propose their own solutions." And then at the end of the day, when they've delivered something, I will validate what they deliver. I will simply make sure that what they deliver met my stated user needs, and between the two we're just doing clarification statements and what that needs to be done. The problem here is that this is likely to result in vendor lock-in. That vendor lock-in will require manager and controller from the same vendor, or I'll have to pay them more to get the rights to that design in order to prevent that vendor lock-in.

That brings us to Option 2, that we have a procurement of the system. So rather than just a bid, now we're talking about more of a proposal. Now we're going to have an integral solution. We're going to explain the user need and define the validation testing. We're going to require delivery of systems engineering documentation. So they have to tell me—they're still going to design it, but they're going to have to tell me the design and release that information.

So that means I get the rights to distribute the documentation to those with a need, to obtain the rights for others to develop products to implement that design, potentially I might want to go as far as saying other competing products. So your signal controller is developing it; I'm going to have another signal controller manufacture and implement it as well, so then I still have a competitive marketplace. And then verify the delivered product implements the design, so that way I make sure that your design documentation agrees with what you actually implemented and get up and running, and then I'll validate, finally, the operation of the delivered product to make sure it meets my user needs.

Now, the challenge here is it's still not standardized. It's still not necessarily a completely open specification, but I might have a limited marketplace. The challenge is, in this case, with the precise wording, I can only distribute it to those who have a need to know. Who has a need to know the detailed design? The manufacturer who developed the design might argue that the only ones who have a need to design it are those with a contract to design it, and signal controller manufacturers might be hesitant to bid on a project if they can't see the full design up front. They're only going to tell them that, "I will give you a design after the contract's been awarded." They might hesitate and increase their costs. So you might still have a limited marketplace, but it's certainly much better than not having that design documentation to begin with.

The third option is a truly open solution. In this case, we explain the user need, define the validation testing. We produce reference systems, engineering documentation, in the public domain ideally, so it's an open solution without any patents, without any other costs involved, and that could be developed by the agency, the manufacturer, system developer, consultant, whoever. The key goal here is at the end of the day that documentation is released in the public domain. That means all of my competitors can access that documentation and know what they're getting into before they submit their bids, and then at the end of the day we verify that the delivered product implements the design and that the design is in fact valid.

Now that might increase your initial costs because whoever's developing this design has to recoup all of their investment up front, rather than saying, "Hey, I can give them a really low bid up front and then recoup my costs later." So it's going to increase your initial costs, but you should benefit longer term.

That brings us to our third question of the day. Which of the following is not true regarding an extension based on an open solution? Documentation is made public; cost of initial deployment may be higher; delivered product needs to be tested against requirements; and likely to result in vendor lock-in. So which of these are not true regarding an extension based on an open solution? Go ahead and make your selection. We'll talk about it.

Well, the correct answer is "likely to result in vendor lock-in". An open solution should not lock you into a single vendor. An open solution prevents true vendor lock-in by ensuring that design is publicly available. Doesn't necessarily mean that everyone's going to implement it, or even more than one vendor implements it, but you're not theoretically locked in because the information is available for anyone else to implement it. Documentation is made public. The fundamental characteristic of an open characteristic is that the documentation is public.

Likewise, vendors are less able to recover costs in subsequent deployments, thereby increasing cost for the initial deployment. So the cost of initial deployment is true for an open solution. So vendors are less able to recover costs, thereby increasing costs for an initial deployment. And then finally, the delivered product needs to be tested against requirements is incorrect. To obtain interoperability enabled by an open solution, the product should be tested against the requirements.

So that brings us to the fourth learning objective: Testing NTCIP 1202 v03. We'll talk a little bit about systems engineering documentation in NTCIP. We'll talk some about the Anaheim case study, and then provide some interim guidance.

So this is the standard V-diagram, used for quite a while now in ITS system deployments. Over on the left-hand side, you see that we start with regional architectures. We then go into projects and you develop a concept of operations, develop system requirements, and then get to the high-level design, detailed design implementation. And then on the upside, you start your unit testing subsystem verification, system verification deployment, and system validation. So different levels of testing on the right side of the V, and then you go into operations maintenance and everything else. In the middle of the V, you see that we are pairing up the different levels of the V. So on the left, you have a concept of operations. You then validate that with a system validation plan.

Likewise, your system requirements are checked during system verification deployment through a system verification plan. What we're focused on is testing our components, which are part of the high-level design and subsystem verification through a subsystem verification plan, also known as subsystem acceptance because often the devices we're procuring are subsystems of a much larger system. The unit device test plan tends to be done by the manufacturer themselves within the detailed design and unit device testing. So at the NTCIP level, we're primarily focused on the integration and everything of making sure my subsystems, my individual components, properly implement the interface specifications that we defined in NTCIP.

Well, the good news is the standard outline for 1200 series documents in NTCIP include not only functional requirements and the design, dialogs, and the MIB, that are connected to each other by this requirements traceability matrix, but then you get down to Annex C, and you

realize Annex C defines your test procedures. Well, that's great news. Now my test procedures are there as well. All I have to do is flip to that annex, I get my test procedures for ASC, and I know how to test my equipment. So if we flip over to Annex C of NTCIP 1202 v03, we see this text.

"It is anticipated that test procedures may be developed as a part of a future revision of NTCIP 1202 v03. Annex C is a placeholder, at present." So not quite as helpful as we had hoped, but there is some good news. So for this, we'll go to a case study. There is a project in the city of Anaheim, California to start looking at this.

They issued in February of 2020 a request for proposals. That project will develop test procedures for all NTCIP 1202 v03 requirements. The vision here is that perhaps—like Purdue University developed something independently and it was incorporated into the standard—the vision here is that if they get this project that USDOT is helping them out on, if they get this project, then hopefully those test procedures will become the future Annex C of 1202. The project is being conducted in two phases per the RFP. Part 1 will focus on the features that were included in v02 or 1202, and then Part 2 of the procedure development will focus on the new features in v03, which are mainly connected vehicles and some other things. The project will actually test three different vendors. So we'll actually have some real-world experience in testing this equipment in a pretty significant segment of the market space. And then it will also provide public domain test software. That's part of the project as well, so that other agencies will be able to pick up the software and test other equipment. And then finally, they will produce a final report.

So in summary, for interim guidance, if you are going to develop a procurement, you need to be looking at testing. It is very expensive to develop your own set of test procedures, because we are talking about a very large document with lots of data that's all interrelated. So rather than developing your own test procedures, what you should be doing is looking at developing a project specification that uses the standardized PRL, that Protocol Requirements List, that we've talked about in Modules A315a and A315b Part 1. Develop from that PRL which features which user needs you need to support for your system, which requirements you need for each one of those user needs, and then we talked about those requirements traced directly to a design, which will then allow test specifications to be developed, and those test specifications will cover—the ones being developed by Anaheim will cover everything that's in the current standard.

So if you do this right and you stick with the standard, now we'll be able to benefit from all of those test procedures that Anaheim is developing. So then you just require compliance to the project PRL and require testing per the test procedures being developed by Anaheim. Now, there is some chicken-and-egg here type issues of timeline considerations. If you start to develop your specification right after this module comes out in 2020, you might actually have to say, "Well, wait for the testing until Anaheim is done," or maybe you give an interim payment or something and wait for final testing until those test procedures are available. I have not seen a final schedule for that Anaheim project. You'll need to contact them directly for that. But you should be able to make your reference to those Anaheim test procedures.

The entire industry is going to be looking towards and forward to those test procedures, so this should be a fairly well-known entity within the industry as it moves forward. Testing should be

performed independently from the manufacturer, which means you can have the agency do the testing; you can have an independent consultant do the test, another agency do the testing for a qualified products list or something; but you don't really want the manufacturer testing their own products. It's much better to have a formal report from an independent tester.

That brings us to our final activity of the day. Which of the below is an appropriate way to test an ASC for conformance to NTCIP 1202 v03? Using testing procedures contained in Annex C of the standard; using Anaheim test procedures when available; connecting to the system seeing if it works; or trusting the vendor. Go ahead and make your answer choices there.

Well, looking at the results, we see that the correct answer is: the Anaheim project aims to develop test procedures for all NTCIP 1202 v03, and those are the procedures that we recommend you reference. Option A would be great in theory. The problem is, as we showed, Annex C of the standard is currently just a placeholder and it doesn't actually contain any test procedures. So you can't really reference Annex C and get anything useful. Connecting to the system and seeing if it works isn't really testing. It might provide some insights as to whether a device will work under normal conditions. More than likely it's not going to test the device under exceptional conditions. That's usually where anomalies arise and you have problems. Finally, trusting the vendor—trust really is not testing at all.

So in summary, we've completed now the entire A315b course, which consisted of two modules, Part 1 and Part 2. Part 1, we looked at identifying the standard requirements, we looked at explaining the purpose and benefits of the Requirements Traceability Matrix and how you customize that to a project-level RTM, which is primarily through extensions and just referencing the standard RTM. We also looked at preparing the ASC specification as a whole.

In this module, Part 2, we looked at the special considerations, both for infrastructure and for functionality, and then we also looked at incorporating requirements not supported by the standardized objects as well as an introduction to testing. The full ASC curriculum then is we started out with A315a. That talked about the user needs and how to fill out their PRL. We've done that. We've done Parts 1 and 2 of A315b, talking about how you deal with the requirements for your implementation.

The next module is T315 that will be looking closer at the test plan as of 2020. There's a current version of that up on the website. That will be updated once the Anaheim test project is a little bit further along, but there is something there to get you started. Right now, as we say, the best thing to do is to look towards the Anaheim test procedures.

So the next course, we'll be looking at recognizing the importance of testing ASCs, applying the rules for developing a sample ASC test plan, which customizes those procedures that Anaheim will be developing to your environment. In the meantime, as I say, as of 2020, the current version that's up there, there's some rules for developing test case specifications and procedures and going through that step-by-step. And then finally, what still will apply even after Anaheim is understanding the test results for NTCIP 1202 v03.

So with that, we thank you for completing this module. Please provide our links for feedback, and thank you for your efforts today, and we look forward to seeing you again on these courses.